European Journal of Operational Research 286 (2020) 673–688

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

A dynamic thompson sampling hyper-heuristic framework for learning activity planning in personalized learning

Ayse Aslan*, Ilke Bakir, Iris F. A. Vis

Department of Operations, University of Groningen, Groningen 9747, AD, the Netherlands

ARTICLE INFO

Article history:
Received 8 September 2019
Accepted 11 March 2020
Available online 19 March 2020

Keywords:
Timetabling
Hyper-heuristics
Dynamic thompson sampling
Personalized learning
OR in education

ABSTRACT

Personalized learning is emerging in schools as an alternative to one-size-fits-all education. This study introduces and explores a weekly demand-driven flexible learning activity planning problem of own-pace own-method personalized learning. The introduced problem is a computationally intractable optimization problem involving many decision dimensions and also many soft constraints. We propose batch and decomposition methods to generate good-quality initial solutions and a dynamic Thompson sampling based hyper-heuristic framework, as a local search mechanism, which explores the large solution space of this problem in an integrative way. The characteristics of our test instances comply with average secondary schools in the Netherlands and are based on expert opinions and surveys. The experiments, which benchmark the proposed heuristics against Gurobi MIP solver on small instances, illustrate the computational challenge of this problem numerically. According to our experiments, the batch method seems quicker and also can provide better quality solutions for the instances in which resource levels are not scarce, while the decomposition method seems more suitable in resource scarcity situations. The dynamic Thompson sampling based online learning heuristic selection mechanism is shown to provide significant value to the performance of our hyper-heuristic local search. We also provide some practical insights; our experiments numerically demonstrate the alleviating effects of large school sizes on the challenge of satisfying high-spread learning demands.

1. Introduction

Education is shifting from traditional one-size-fits-all models which offer standardized learning paths for everyone in a certain group (e.g., age, level) to personalized learning (Bray & McClaskey, 2013; West-Burnham & Coates, 2005). Schools are implementing various personalized learning models in which students have the freedom to customize their learning paths and learn at their own pace with their own methods throughout the world (see Eiken (2011), Prain et al. (2013) and Kannan, van den Berg, and Kuo (2012), for examples from Europe, Australia, and U.S.A., respectively). In Europe, the Swedish kunskapsskolan personalized learning model, initiated in 2000 in four schools in Sweden, is now being implemented in more than 100 schools around the world (see http://www.kunskapsskolan.com/thekednetwork). According to a report by the European Commission published in 2017 (see https://ec.europa.eu/epsc/publications/other-publications/10-trends-transforming-education-we-know-it_en), personalized learning is an important strategic trend to transform education.

Contrary to one-size-fits-all, students in personalized learning are not tied to classes, instead their learning needs are regarded individually. In personalized learning technology plays an important role; online learning portals are often available for students so that they can also learn independently through self-study learning activities in schools. This gives students the freedom to choose their learning methods. In personalized learning, students are the directors of their own learning processes; they set their own goals, with the support of learning coaches, and actively demand learning activities to reach them. An important task for schools is to satisfy students' learning demands on time by planning relevant in-class and self-study learning activities with the utilization of teachers and classrooms.

In these own-method own-pace personalized learning models in which many students may demand many different activities at any time, neither students nor teachers are tied to fixed groups. Activity groups are formed flexibly each time by flexibly grouping student demands in activities and allocating suitable teachers

* Corresponding author.
E-mail addresses: ayse.aslan@rug.nl (A. Aslan), i.bakir@rug.nl (I. Bakir), i.f.a.vis@rug.nl (I.F.A. Vis).

https://doi.org/10.1016/j.ejor.2020.03.038

and classrooms to activities. For example, a student may be with different groups of students and also with different teachers in learning activities. This study introduces *the weekly flexible demand-driven learning activity planning problem* of personalized learning in which schools plan learning activities flexibly each week based on student demands. Due to the lack of fixed groups, this problem involves decisions on individual student demands and also on school resources (i.e., teachers, classrooms and time blocks).

Our problem partially relates to the educational timetabling problems due to common elements such as students, teachers and classrooms, and also due to common constraints such as scheduling conflicts, availability and capacity constraints. Educational timetabling problems are extensively studied (see Pillay, 2014; Pillay, 2016; Schaerf, 1999 for reviews). Many of these problems are computationally intractable, either $\mathcal{NP}$-complete or $\mathcal{NP}$-hard problems. Various approaches such as single-solution local search methods (Fonseca & Santos, 2014), population-based search methods (Beligiannis, Moschopoulos, Kaperonis, & Likothanassis, 2008; Santiago-Mozos, Salcedo-Sanz, DePrado-Cumplido, & Bousoño-Calzón, 2005), hyper-heuristics (Ahmed, Ozcan, & Kheiri, 2015; Pillay & Banzhaf, 2009), matheuristics (Dorneles, de Araújo, & Buriol, 2014), integer programming techniques (Fonseca, Santos, Carrano, & Stidsen, 2017; Phillips, Waterer, Ehrgott, & Ryan, 2015) and graph-theoric approaches (Kannan et al., 2012) are proposed and tested for these problems. Although the studied problems in the literature are mostly concentrated on traditional educational models, there are a few studies which explore student-centred planning and timetabling problems. In Santiago-Mozos et al. (2005), a student-preference based course timetabling problem in a Spanish university is presented. Also, a more recent study (Kristiansen, Sørensen, & Stidsen, 2011) presents the student-centric elective course planning problem of Danish high schools. Kannan et al. (2012) has proposed a multi-stage graph-theoric approach to the scheduling problem of a group of personalized learning schools in New York City. Implementations of personalized learning differ in terms of their degree of freedom offered to students. In Santiago-Mozos et al. (2005), Kristiansen et al. (2011) and Kannan et al. (2012), students are only given the freedom to customize their learning paths, their curricula, by providing preferences over a set of courses at the beginning of a semester or a year. Our problem, on the other hand, considers models in which students are also given the freedom to progress at their own pace with their own methods for every learning activity of their courses anytime.

The studied educational timetabling problems usually consider traditional educational models and therefore study the assignment of predetermined events (e.g., course-class meetings in school timetabling (Pillay, 2014)) to available times. The main distinguishing aspect of our problem is that learning activities are not predetermined; they are planned based on student demands. Namely, which learning activities to plan and how many sessions of each learning activity to plan in a week are also decisions to be made in our problem. Thus, we classify this problem as a demand-driven planning problem rather than a timetabling problem.

The dynamic, flexible, and demand-driven nature of this planning problem provides opportunities to learn from and contribute to the state-of-the-art in the area of logistics; more specifically, warehouse order picking (de Koster, Le-Duc, & Roodbergen, 2007), dynamic vehicle routing (Pillac, Gendreau, Gueret, & Medaglia, 2013), train routing and scheduling (Cordeau, Toth, & Vigo, 1998), among many others. For example, modern warehouse order picking problems face the challenge of dynamic order arrivals due to the growth of e-commerce. Dynamically arriving orders must be batched in pick lists and then picked by order pickers as efficiently as possible (de Koster et al., 2007). Methodologically, parallels can be drawn between creating pick lists in this problem and creating activity groups by batching student demands in our problem. We get inspiration from and strive to provide insights for these complex and dynamic logistics problems, which can directly benefit from approximate systematic solution methods such as ones we propose in this paper.

As also argued in Kannan et al. (2012), due to the freedom offered to students, finding good solutions becomes a computational challenge as solution space grows. Therefore, this task is not suitable for simple solution strategies such as rules of thumb, instead, systematic planning tools are necessary to explore the solution space and find high-quality solutions. This paper aims to present an efficient systematic method for the flexible demand-driven planning problem. We demonstrate the applicability of our proposed method in the context of secondary schools in the Netherlands.

The remainder of the paper is organized as follows. Section 2 describes the weekly activity planning problem. Section 3 presents the MILP formulation. Section 4 presents the proposed heuristic approaches. Section 5 gives the computational experiments, which provide performance analysis of the proposed solution approaches. Section 6 provides practical insights and decision support for schools. Lastly, Section 7 concludes the paper.

2. Problem description

The weekly flexible demand-driven learning activity planning problem relates to schools which implement own-pace own-method personalized learning models. The learning activity planning in these models is initiated when students demand learning activities for the lesson units of the learning goals of their courses. Fig. 1a illustrates the mechanism of learning activity planning in these models, as opposed to the mechanism of one-size-fits-all models which is depicted in Fig. 1b.

The main distinguishing aspect of planning in personalized learning compared to traditional education is the lack of fixed groups (i.e., classes). Yearly or semesterly produced weekly cyclic timetables are not appropriate in personalized learning, as learner groups are dynamically formed each time based on varying learning demands. In personalized learning, students can be grouped in learning activities flexibly and teachers and classrooms can be allocated flexibly as well. The planning problem of the schools here is to weekly plan in-class and self-study learning activities by flexibly composing activity groups and allocating resources to satisfy students' learning demands.

The formal description of the problem with notation (see Table 1) is as follows.

In personalized learning schools, a set of in-class activities $a \in A$ are offered to a set of students $s \in S$. Each in-class learning activity $a \in A$ is uniquely associated to a lesson $l \in L$ of a learning goal $g \in G$ of a course $c \in C$. At the end of every week, students provide their high priority A_s^1 and low priority A_s^2 demands for the in-class activities that they would like to be planned for the following week. Their school plans a number of sessions of the demanded in-class learning activities in a set of time blocks $b \in B$ of a set of school days $d \in D$ for the next week by flexibly assigning teachers, classrooms and students to the sessions. All demands should be satisfied as much as possible with the efficient use of school resources. However, if possible, the number of activities on course c that are assigned to student s in day d should not be exceeding a daily course limit of CL for each day $d \in D$ for each course $c \in C$ and for each student $s \in S$. Any in-class activity session takes one time block. In personalized learning schools, self-study activities with available learning materials, such as online learning portals, can stand as an alternative to in-class learning. Therefore, usually

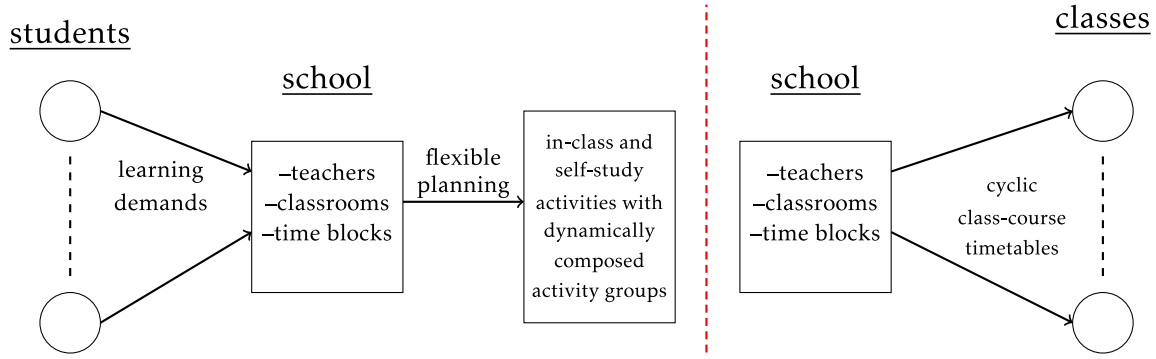

Fig. 1. Personalized learning versus traditional education.

Table 1
Input notation.

Sets	
	$a \in A$: set of in-class activities, $s \in S$: set of students, $c \in C$: set of courses,
	$g \in G$: set of learning goals, $l \in L$: set of lessons, $t \in T$: set of teachers,
	$r \in R$: set of classrooms, $d \in D$: set of weekdays, $b \in B$: set of time blocks
Subsets	
	A_s^1 : set of in-class activities demanded with high priority by student s
	A_s^2: set of in-class activities demanded with low priority by student s
	A_c: set of activities on course c
	A_c^1: set of in-class activities on course c which can be only taught by first-level teachers
	A_a: set of in-class activities that precede activity a
	A^{ne}: set of in-class activities on conventional courses
	C^e: set of non-conventional courses
	R_c^e: set of classrooms of non-conventional course c
	R^{ne}: set of classrooms of conventional courses
	T_{bd}: set of teachers available in block b of day d
	T_c: set of teachers of course c
	T_c^1: set of first-level teachers of course c
	B_c: set of time blocks that are preferred by course c
Parameters	
	CL: daily course limit of students
	K_a: classroom capacity of in-class activity a
	WT: weekly in-class assignment limit of teachers
	SE: maximum number of students that a teacher can guide in self-study

a large self-study environment is available in schools for self-study learning activities.

In-class activity sessions take place in a set of classrooms $r \in R$. Some non-conventional courses $c \in C^e$ such as Physical Education, Art and Information Technologies can only be accommodated in equipped classrooms R_c^e such as gyms, art studios and computer labs. The remaining conventional courses $c \in C - C^e$ can be accommodated in traditional classrooms R^{ne} with no special equipment. The classroom capacities can be defined for activities without knowing explicitly in which classrooms they are going to be planned (see Assumption A1). The capacities of classrooms K_a set a limit on the number of students that can be grouped in sessions of in-class activities $a \in A$.

Sessions are taught by a set of teachers $t \in T$. Each session should be assigned to a teacher. Some teachers work part-time; in any time block $b \in B$ of any day $d \in D$ only some teachers $t \in T_{bd}$ are available. Secondary school teachers in the Netherlands have two teaching qualification levels: first and second level. The first-level teachers T_c^1 of course $c \in C$ are qualified to teach any lesson of any learning goal of course c. The second-level teachers $T_c - T_c^1$ of course $c \in C$, on the other hand, are qualified to teach the lessons of the majority of learning goals, but not the ones of the higher levels $a \in A_c^1$. It is not desired that teachers are assigned to in-class activity sessions in a week more than a weekly limit of WT. Apart from in-class activities, teachers are also assigned to self-study environment in order to guide self-study activities of students. Any teacher $t \in T$ can provide guidance for up to SE many students' self-study activities. Sufficient teacher levels should be assigned for self-study activities as much as possible in any time block. Balancing teachers' workloads in a week is desired in both in-class and self-study activity assignments. For a teacher, the in-class (self-study) activity workload in a week is measured by the utilization rate of his/her total available time with in-class (self-study) activities. Teachers T_c of each course $c \in C$ desire to have similar in-class activity workloads as much as possible among themselves. For the self-study activities, all teachers T, regardless of their courses, desire to have similar self-study workloads.

Lastly, schools can indicate preferred time blocks B_c for courses to plan in-class activities relating to courses. For example, for courses that require high concentration levels, schools may provide preference of morning time blocks for the activities relating to these courses.

This problem seeks to produce a weekly learning activity plan at the end of each week, for the coming week, by deciding on how many sessions of each activity to plan (x_{abd}), which students to assign to the planned activities (z_{sabd}) and which resources to assign to the planned in-class (y_{tbd}) and self-study activities (y'_{tbd}). Note that classroom assignment decisions are left out (see Table 2), and teachers are only assigned to in-class/self-study states without being assigned to specific sessions of learning activities, also students are not explicitly assigned to specific activity sessions; they are only assigned to activities. With the flexibility in forming learning

Table 2
Decision variables

Variable	Description
$z_{sabd} \in \{0, 1\}$	1 if student $s \in S$ is assigned to in-class activity $a \in A_s^1 \cup A_s^2$ in block $b \in B$ of day $d \in D$, 0 otherwise
$x_{abd} \in \mathbb{N}$	number of sessions planned of in-class activity $a \in \bigcup_{s \in S} A_s^1 \cup A_s^2$ in block $b \in B$ of day $d \in D$
$y_{tbd} \in \{0, 1\}$	1 if teacher $t \in T$ is assigned to an in-class activity session in block $b \in B$ of day $d \in D$, 0 otherwise
$y'_{tbd} \in \{0, 1\}$	1 if teacher $t \in T$ is assigned to self-study environment in block $b \in B$ of day $d \in D$, 0 otherwise

activity sessions, teachers do not have preferences over the in-class activities that they are assigned, students do not have preferences over the teachers of the sessions and over the students that they are assigned together to the sessions, and activity sessions do not have preferences over the classrooms. This allows a significant reduction in decision layers of the problem. The reduced decisions can be reconstructed by a post-processing procedure without compromising optimality. One such procedure is given in the online supplement.

Below we list our assumptions relating to the use of teachers and classrooms in this planning problem. These assumptions are confirmed by an expert to be mostly realistic in the case of the secondary schools in the Netherlands.

- A1 : Each traditional classroom is identical to any other traditional classroom and each equipped classroom of a non-conventional course is identical to any other classroom of the same course. This assumption establishes a direct link between x_{abd} and z_{sabd} without the knowledge of classroom assignments. In fact, this assumption is also necessary for the decision reduction made on the classroom assignments.
- A2: When students are not assigned to in-class activities, they are doing self-study activities in the self-study environment.
- A3: The size of the self-study environment is large enough to accommodate all students in a school. Therefore, physical space assignments for self-study activities are not included in the problem.
- A4: Each teacher $t \in T$ is specialized in only one course $c \in C$. Note that, this assumption is not necessary for the mathematical model. However, it simplifies the calculations in the feasibility checking phases of our constructive heuristics (see Section 4.2) and makes the course based in-class workload imbalance calculations (see SC2) more meaningful.

The described problem seeks the desired weekly activity plan that satisfies all described hard constraints and violates as few soft constraints as possible.

3. MILP formulation

The plan that is to be produced must satisfy the following hard constraints.

- HC1: A student can not be assigned to more than one activity session at any time.

$$\sum_{a \in A} z_{sabd} \leq 1 \qquad \forall\, s \in S,\ \forall\, b \in B,\ \forall\, d \in D \tag{1}$$

- HC2: A student can only be assigned to a session of an activity that (s)he demands.

$$\sum_{b \in B} \sum_{d \in D} z_{sabd} \leq \mathbb{1}_{\{a \in A_s^1\}} + \mathbb{1}_{\{a \in A_s^2\}} \qquad \forall\, a \in A,\ \forall\, s \in S \tag{2}$$

- HC3: The number of students assigned to activity sessions must respect classroom capacities.

$$\frac{\sum_{s \in S} z_{sabd}}{K_a} \leq x_{abd} \qquad \forall\, a \in A,\ \forall\, b \in B,\ \forall\, d \in D \tag{3}$$

- HC4: A suitable teacher must be assigned to each in-class activity session.

$$\sum_{a \in A_c^1} x_{abd} \leq |T_{bd} \cap T_c^1| \qquad \forall\, c \in C,\ \forall\, b \in B,\ \forall\, d \in D \tag{4}$$

$$\sum_{a \in A_c} x_{abd} \leq |T_{bd} \cap T_c| \qquad \forall\, c \in C,\ \forall\, b \in B,\ \forall\, d \in D \tag{5}$$

$$\sum_{t \in T_c^1 \cap T_{bd}} y_{tbd} \geq \sum_{a \in A_c^1} x_{abd} \qquad \forall\, c \in C,\ \forall\, b \in B,\ \forall\, d \in D \tag{6}$$

$$\sum_{t \in T_c \cap T_{bd}} y_{tbd} = \sum_{a \in A_c} x_{abd} \qquad \forall\, c \in C,\ \forall\, b \in B,\ \forall\, d \in D \tag{7}$$

Constraints (6) and (7) would be sufficient for satisfying HC4. However, constraints (4) and (5) are valid inequalities that provide a tighter LP relaxation. They are also used in the decomposition heuristic later (see Section 4.2.1).

- HC5: A teacher can not be assigned to more than one activity at any time and (s)he can only be assigned if (s)he is available.

$$y'_{tbd} + y_{tbd} \leq \mathbb{1}_{\{t \in T_{bd}\}} \qquad \forall\, t \in T,\ \forall\, b \in B\,, \forall\, d \in D \tag{8}$$

- HC6: A suitable classroom must be assigned to each in-class activity session.

$$\sum_{a \in A_c} x_{abd} \leq |R_c^e| \qquad \forall\, c \in C^e,\ \forall\, b \in B,\ \forall\, d \in D \tag{9}$$

$$\sum_{a \in A^{ne}} x_{abd} \leq |R^{ne}| \qquad \forall b \in B,\ \forall\, d \in D \tag{10}$$

Note that these constraints are only for making sure that the activity decisions are taken such that they will be feasible with respect to classroom resources. Specific classroom assignments are done via a post-processing procedure.

- HC7: Lessons of learning goals have precedence relations; students must follow them in the right order.

$$z_{sabd} \leq \frac{\sum_{a' \in A_a} \sum_{(b',d') < (b,d)} z_{sa'b'd'}}{\sum_{a' \in A_a} \mathbb{1}_{\{a' \in A_s^1 \cup A_s^2\}}} \quad \forall\, s \in S,\ \forall\, b \in B, \forall\, d \in D,$$
$$\forall\, a \in A_s^1 \cup A_s^2 \quad s.t. \quad |(A_s^1 \cup A_s^2) \cap A_a| \neq 0 \tag{11}$$

The notation $(b', d') < (b, d)$ is used to denote all blocks (b', d') that precede time block b of day d.

In addition to the presented hard constraints, there are several soft constraints regarding the quality of learning activity plans. These soft constraints do not have to be satisfied, however they are desired to be satisfied as much as possible. For each soft constraint an auxiliary variable is defined (see Table 3). These variables measure the violations of each soft constraint.

- SC1: Satisfying high and low priority student demands.

$$\alpha_s^1 \geq \sum_{a \in A_s^1} \left(1 - \sum_{b \in B} \sum_{d \in D} z_{sabd}\right) \qquad \forall\, s \in S \tag{12}$$

$$\alpha_s^2 \geq \sum_{a \in A_s^2} \left(1 - \sum_{b \in B} \sum_{d \in D} z_{sabd}\right) \qquad \forall\, s \in S \tag{13}$$

Table 3
Auxiliary variables.

Variable	Description
$\alpha_s^1 \in \mathbb{N}$	number of unmet high priority demands of student $s \in S$
$\alpha_s^2 \in \mathbb{N}$	number of unmet low priority demands of student $s \in S$
$\alpha_c^3 \in \mathbb{R}_0^+$	extent of imbalance in the in-class activity workloads among teachers in T_c
$\alpha_c^4 \in \mathbb{N}$	number of times that sessions of in-class activities in A_c are planned in $B - B_c$
$\alpha_{scd}^5 \in \mathbb{N}$	extent of violating daily course limit of CL for student $s \in S$ for course $c \in C$ in day $d \in D$
$\alpha_t^6 \in \mathbb{N}$	extent of violating weekly in-class assignment limit of WT for teacher $t \in T$
$\alpha_{bd}^7 \in \mathbb{N}$	number of shortage teachers in self-study environment in time block $b \in B$ of day $d \in D$
$\alpha^8 \in \mathbb{N}$	extent of imbalance in the self-study activity workloads among teachers T
$\alpha^9 \in \mathbb{N}$	number of planned in-class activity sessions in a week

- SC2: Balancing teachers' in-class workloads.

$$\alpha_c^3 = \beta_c^{max-class} - \beta_c^{min-class} \quad \forall c \in C,$$
$$where \quad \beta_c^{max-class}, \beta_c^{min-class} \in \mathbb{R}_0^+$$
$$\beta_c^{min-class} \leq \frac{\sum_{b\in B}\sum_{d\in D} y_{tbd}}{\sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}}}$$
$$\forall\, t \in T_c \; s.t. \sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}} > 0, \; \forall\, c \in C$$
$$\beta_c^{max-class} \geq \frac{\sum_{b\in B}\sum_{d\in D} y_{tbd}}{\sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}}}$$
$$\forall\, t \in T_c \; s.t. \sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}} > 0, \; \forall\, c \in C \tag{14}$$

The method employed here for measuring the imbalance is the simple method of taking the difference between maximum and minimum workloads.

- SC3: Planning activity sessions in their preferred time blocks.

$$\alpha_c^4 = \sum_{a\in A_c}\sum_{d\in D}\sum_{b\notin B_c} x_{abd} \qquad \forall\, c \in C \tag{15}$$

- SC4: Limiting the extent of exceeding students' daily course limit of CL.

$$\alpha_{scd}^5 \geq \sum_{a\in A_c}\sum_{b\in B} z_{sabd} - CL \qquad \forall\, s \in S, \; \forall\, c \in C, \forall\, d \in D \tag{16}$$

- SC5: Limiting the extent of exceeding teachers' weekly in-class assignment limit of WT.

$$\alpha_t^6 \geq \sum_{b}\sum_{d} y_{tbd} - WT \qquad \forall\, t \in T \tag{17}$$

- SC6: Assigning required numbers of teachers to self-study at any time.

$$\alpha_{bd}^7 \geq \frac{\sum_{s\in S}(1 - \sum_{a\in A} z_{sabd})}{SE} - \sum_{t\in T} y'_{tbd} \qquad \forall\, b \in B, \; \forall\, d \in D \tag{18}$$

- SC7: Balancing teachers' self-study workloads.

$$\alpha^8 = \beta^{max-self} - \beta^{min-self}, \quad where \;\; \beta^{max-self}, \beta^{min-self} \in \mathbb{R}_0^+$$
$$\beta^{min-self} \leq \frac{\sum_{b\in B}\sum_{d\in D} y'_{tbd}}{\sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}}} \quad \forall\, t \in T \; s.t. \sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}} > 0$$
$$\beta^{max-self} \geq \frac{\sum_{b\in B}\sum_{d\in D} y'_{tbd}}{\sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}}} \quad \forall\, t \in T \; s.t. \sum_{b\in B}\sum_{d\in D} \mathbb{1}_{\{t\in T_{bd}\}} > 0 \tag{19}$$

- SC8: Minimizing the number of sessions planned in a week.

$$\alpha^9 = \sum_{b\in B}\sum_{d\in D}\sum_{a\in A} x_{abd} \tag{20}$$

This constraint imposes the efficiency in planning.

We define for each penalty auxiliary variable $\alpha_*^n, n = 1, 2, ..., 9$ a weight parameter $w^n \in \mathbb{R}^+, n = 1, 2, ..., 9$ and formulate the following cost function, which is the weighted sum of violations of the soft constraints of our problem.

$$MIN \quad \sum_{n=1}^{2}\sum_{s\in S}\alpha_s^n w^n + \sum_{n=3}^{4}\sum_{c\in C}\alpha_c^n w^n + \sum_{s\in S}\sum_{c\in C}\sum_{d\in D}\alpha_{scd}^5 w^5 + \sum_{t\in T}\alpha_t^6 w^6 + \sum_{b\in B}\sum_{d\in D}\alpha_{bd}^7 w^7 + \sum_{n=8}^{9}\alpha^n w^n \tag{21}$$

We show that the weekly flexible demand-driven learning activity planning problem is $\mathcal{NP}$-hard. In fact, it can be shown that many of the educational timetabling problems, which are already proven intractable in the literature, are special cases of this problem. For instance, the student scheduling problem, which is proven $\mathcal{NP}$-hard by Cheng, Kruk, and Lipman (2002), only assigns students to course sections (can be thought as the activity sessions of our problem) to fulfill student demands and therefore is a special case of our problem, where sessions of activities already have been planned and assigned to times, teachers and classrooms.

4. Dynamic thompson sampling hyper-heuristic framework

The MILP model described in the previous section is not solvable by the state-of-the-art solvers within reasonable times for moderate sized instances. Consequently, we consider an approximate approach and develop a hyper-heuristic framework for producing solutions for this intractable problem. Hyper-heuristics, which are heuristic search methods that use heuristic methods to choose from a pool of simpler (low-level) heuristics, are already in use to solve many computationally intractable educational timetabling problems (Ahmed et al., 2015; Burke, McCollum, Meisels, Petrovic, & Qu, 2007; Pillay & Banzhaf, 2009). The use of hyper-heuristics in educational timetabling has recently been reviewed by Pillay (2016). This section presents our dynamic Thompson sampling single-solution selection hyper-heuristic framework. The overall solution methodology is summarized in Fig. 2.

Differently from typical educational timetabling problems, this planning problem contains a very large number of decision dimensions. This fact enables many opportunities for different solution methods and strategies. For instance, there are many ways of encoding solutions and performing search on various spaces. With the motivation of exploring many solution approaches, here we describe two different constructive heuristics which we use as initial solution generators for our hyper-heuristic local search. In fact, in the online supplement, we also briefly discuss other heuristics that we test for this problem; these also include genetic algorithms which explore different solution representations. Also with the involvement of numerous soft constraints in our problem, exactly nine many, the number of low-level heuristics in our hyper-heuristic framework is significantly higher than that of a typical educational timetabling problem.

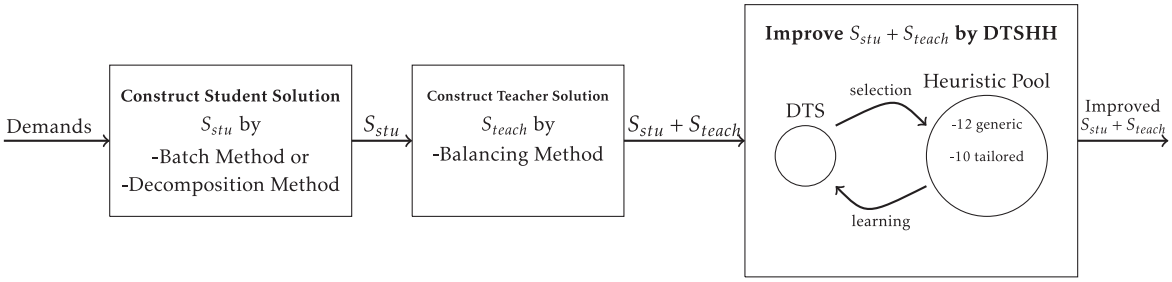

Fig. 2. Solution methodology.

4.1. Solution encoding

We use direct encoding which maps each student $s \in S$ to one of the in-class activities in $A_s^1 \cup A_s^2$ or to self-study, which we call "student solution", and each teacher $t \in T$ to in-class or self-study assignment, or to the idle state, which we call "teacher solution", for every time block $b \in B$ of every day $d \in D$. The number of sessions planned of each in-class activity $a \in A$, x_{abd}, is determined indirectly from the student solution as classrooms of activities K_a have fixed sizes.

4.2. Initial Solutions

Our hyper-heuristic framework applies local search on a feasible good-quality initial solution. Due to the fact that our problem involves many decisions, it is difficult to construct a solution heuristically in an integrative way. In this study, a feasible initial solution is obtained in two phases with greedy constructive heuristics. Firstly, a feasible initial student solution is built and then the generated student solution is used to construct the corresponding feasible teacher solution, to fulfill teacher assignment requirements for the activities planned with student assignments. Two distinct constructive heuristics are utilized to generate good-quality initial solutions. These heuristics differ in the first phase of generating student solutions.

A student solution can always be generated in a feasible way with respect to teacher assignment requirements of activity sessions, without the explicit decisions on teacher assignments (y_{tbd}, y'_{tbd}). Specifically, this feasibility relates to HC4. With assumption A4, the computational effort of checking this feasibility is not significant at all; only the available teacher levels in each course need to be in line with the number of sessions planned of that course; constraints (4) and (5) are sufficient to check this feasibility, without the need of constraints (6) and (7). However, this decomposition of the solution into student and teacher solutions will affect the solution quality as there are some dependencies between student and teacher solutions. Namely, the firstly built student solution will directly affect the quality in the self-study environment, concerning the teacher shortages in the environment, and also the teachers' weekly workloads and their workload imbalances. The proposed heuristics build student solutions before teacher solutions because meeting student demands is prioritized over other soft constraints in practical instances. Besides, the construction phase is only the first phase of our approach; the local search applied after this phase is able to improve a solution with respect to the teacher-related quality metrics, as local search is made in an integrative manner on both student and teacher solutions.

4.2.1. Student solutions

This section describes the two methods that are used to generate feasible student solutions. The detailed pseudocodes of the methods can be found in the online supplement.

Batch method: This heuristic plans a feasible in-class session by selecting a learning activity and an available time block of a day, then a number of students, up to the classroom capacity of the selected activity, who can feasibly be assigned to the selected activity at the selected time are assigned in a greedy fashion, at each iteration. The priorities of the activities to be selected at iterations are determined based on their potential to reduce the costs which relate to the student solution (costs due to $\alpha_s^1, \alpha_s^2, \alpha_c^4$ and α_{scd}^5). This procedure stops when there are no possibilities to plan feasible sessions. The feasibility of planning a new session of a learning activity is determined by the levels of the suitable teachers and classrooms of the course of the activity (by checking constraints (4) and (5) and constraints (9) and (10)).

Decomposition method: This heuristic decomposes the student solution part of the MILP model per time block per day. The subproblem of each time block of each day is simple enough to be solved by Gurobi to almost optimality quickly. This simplification is mostly due to the fact that the subproblem does not contain the computationally challenging lesson precedence constraints (12). Each subproblem consists of constraints (1)–(5), (8)–(10), (12) and (13), (15) and (16) and (20) with reduced time block and day dimensions and an objective function that considers only the student solution-related quality measures in (21).

4.2.2. Teacher solutions

A workload balancing constructive heuristic is used to build a teacher solution from a student solution. In this heuristic, initially all teachers are considered idle in every time block of every day. For every block $b \in B$ of each day $d \in D$, firstly the teacher assignments for sessions of in-class activities that require first-level teachers are made. For each course $c \in C$, only first-level available teachers, $t \in T_{bd} \cap T_c^1$, are considered. In this process, teachers who are least assigned to in-class activities are prioritized. This is to balance teachers' in-class activity workloads. When assignments for the sessions of activities in A_c^1 are made, assignments are made for the sessions of remaining activities on course $c \in C$ in the same way. Note that an assigned teacher is not considered available anymore for later assignments in the same time block. This is repeated for every course $c \in C$, and then, lastly, a number of teachers who are still available are assigned for self-study activities. This number is based on *SE* and the number of students who are not assigned to any in-class activities in time block b of day d. Assignments for self-study are also performed in a similar fashion where teachers' current self-study workloads are considered for prioritizing teachers. This procedure is repeated until teacher assignments are made for every block $b \in B$ of each day $d \in D$. The pseudocode of this heuristic is also given in the online supplement.

4.3. Dynamic thompson sampling local search hyper-heuristic

We develop a single-solution hyper-heuristic framework which uses the dynamic multi-armed bandit algorithm of dynamic

Thompson sampling to improve a constructed initial greedy solution. Our framework works as a local search method on a single solution with a pool of predefined low-level heuristics that act as neighborhood structures. Single-solution local search selection hyper-heuristics perform search on a single solution with heuristic selection and move acceptance processes until a stopping criterion is met (Burke et al., 2009). The heuristic selection process at each iteration selects a low-level heuristic from the pool to apply on the current solution. After a candidate solution is found by applying the selected low-level heuristic on the current solution, the move acceptance process decides whether to accept or reject the candidate solution.

In our framework, the deterministic acceptance criterion of accepting only non-worsening solutions is selected. It is true that only accepting good solutions limits the scope of the search space and move acceptance strategies such as simulated annealing or threshold acceptance that also accept some worsening solutions can be useful for escaping local optima. However, our investigations could not find the benefits of using these acceptance strategies in this problem. This is likely to be related to the issue that the search space of this problem is extremely large that it may take tremendous computational effort for these strategies to make a fine exploration of the search space. Therefore, we limit ourselves to local optima solutions. However, it is important to note that this does not lead to a myopic search, since our framework consists 22 neighbourhoods.

Hyper-heuristic frameworks that use learning to guide the heuristic selection process use historical performances of low-level heuristics as guidance. When learning takes place during the search process of an instance, frameworks are classified as online learning hyper-heuristics (Burke et al., 2009). Here, a dynamic Thompson sampling-based online learning mechanism is presented to guide the heuristic selection process, to select an appropriate low-level heuristic at each iteration of the search. The dynamic Thompson sampling (DTS) algorithm, which is introduced by Gupta, Granmo, and Agrawala (2011) to solve dynamic multi-armed bandit problems, is integrated in the heuristic selection process of our framework.

We arrived at integrating this learning algorithm in our framework by recognizing the parallelism between “the search game”, selecting a heuristic at each iteration to reach a good solution at the end of the search in dynamic search spaces, and “the gambler’s game”, selecting an arm to pull at each step to reach a state with a high reward in dynamic environments (for a similar parallelism see Fialho, Da Costa, Schoenauer, & Sebag (2010)). Multi-armed bandit problems are concerned with the balance of exploitation and exploration in games. The typical multi-armed bandit problem of static environments considers that a single player, the so-called gambler, chooses an arm to pull from a given set of arms which are associated with unknown probabilistic reward mechanisms at each step in a sequential game, in order to maximize his/her total expected reward at the end of the game. The gambler learns about the reward distributions of the arms as time passes, in an online fashion, which (s)he can exploit for the next steps of the game. However, the gambler may also choose to increase his/her knowledge of the reward mechanisms of the arms by exploring. In the static version, the reward mechanisms of the arms do not change in time such that there is a best arm that the gambler wants to discover. However, in the search games of heuristics there is not a single heuristic/operator that would be best for any time (Da Costa, Fialho, Schoenauer, & Sebag, 2008) for any solution. Hence, the dynamic version of multi-armed bandit problem is more suitable for building parallelism for the search game. Many dynamic versions of multi-armed bandit problem algorithms such as Upper Confidence Bound (UCB) bandit algorithm are already tested (Fialho et al., 2010) for guiding the search processes. In this study, the performance of the DTS algorithm in Gupta et al. (2011) as a heuristic selector is tested to explore more on how dynamic multi-armed bandit based algorithms perform as operator/heuristic selectors in search algorithms.

The DTS algorithm in Gupta et al. (2011) is introduced for dynamic bandit problems in which reward probabilities of the beta-Bernoulli arms are Brownian motion processes. This algorithm is an order statistics-based Thompson sampling which tracks dynamic changes in reward probabilities with an exponential filtering technique. Gupta et al. (2011) demonstrates that the DTS algorithm outperforms Thompson sampling and two UCB based algorithms for dynamic bandits. Our framework considers a beta-Bernoulli bandit for the heuristic selection process; rewards of low-level heuristics follow Bernoulli distributions and reward success probabilities of heuristics follow beta distributions. When a low-level heuristic improves the current solution, it is considered a success and the heuristic is rewarded. This mechanism does not consider the extent of improvements. Our choice is deliberate to give fairer chances to the low-level heuristics. For instance, some low-level heuristics that act on the teacher solutions, although not having greater chances of improving the current solution to a great extent immediately, create high-improvement opportunities for the succeeding low-level heuristics that act on student solutions. Our heuristic selection process uses the expectation values of beta-distributed reward success probabilities of low-level heuristics.

Algorithm 1 presents the pseudocode of our framework of dynamic Thompson sampling based single solution hyper-heuristic (DTSHH). This framework uses three parameters that relate to the DTS algorithm. C^{DTS} denotes a threshold value that reflects for how long to postpone the tracking of changes in reward probabilities. The remaining parameters are initialization of beta distribution parameters α^k and β^k for each low-level heuristic k. Note that when C^{DTS} is sufficiently large, our heuristic selection process will behave as a traditional Thompson sampling algorithm, which does not track the changes at all, by only using the first set of parameter update rules, i.e., lines 18–19 of Algorithm 1.

Algorithm 1 Pseudocode of the dynamic Thompson sampling hyper-heuristic (DTSHH) framework.

1: Initialize C^{DTS} and α^k, β^k, for $k = 1 : N$; $Pool \leftarrow \{LLH_1, \ldots, LLH_N\}$;
2: $S_{current} \leftarrow$ *Generate Initial Greedy Solutions*;
3: $f_{current} \leftarrow$ *Calculate Objective*$(S_{current})$;
4: **while** time & iteration limit not reached **do**
5: $h \leftarrow Find_{k \in 1 \ldots N} \;\; s.t. \;\; \frac{\alpha^k}{\alpha^k+\beta^k} = \max_{n \in 1,\ldots N} \frac{\alpha^n}{\alpha^n+\beta^n}$;
6: $reward \leftarrow 0$;
7: $S_{candidate} \leftarrow Apply(S_{current}, LLH_h)$;
8: $f_{candidate} \leftarrow$ *Calculate Objective*$(S_{candidate})$;
9: **if** $f_{candidate} \leq f_{current}$ **then**
10: $S_{current} \leftarrow S_{candidate}$; $f_{current} \leftarrow f_{candidate}$;
11: **if** $f_{candidate} < f_{current}$ **then**
12: $reward \leftarrow 1$;
13: **end if**
14: **end if**
15: **if** $\alpha^h + \beta^h < C^{DTS}$ **then**
16: $\alpha^h \leftarrow \alpha^h + reward$; $\beta^h \leftarrow \beta^h + (1 - reward)$;
17: **else**
18: $\alpha^h \leftarrow (\alpha^h + reward)\frac{C^{DTS}}{C^{DTS}+1}$; $\beta^h \leftarrow (\beta^h + (1 - reward))\frac{C^{DTS}}{C^{DTS}+1}$;
19: **end if**
20: **end while**
21: **return** $S_{current}$

We classify our low-level heuristics in two groups: (1) *generic* and (2) *tailor-made* low-level heuristics. We have 12 *generic* low-level heuristics which are often used in local-search (perturbative) selection hyper-heuristics in solving various educational timetabling problems (Pillay, 2016). These heuristics apply simple mutation (e.g., move and swap) and hill-climbing operations to perturb a current solution.

Additionally to these, 10 *tailor-made* low-level heuristics are developed which aim to reduce violations of specific soft constraints of our planning problem. Our tailor-made low-level heuristics deliberately guide to neighbourhoods that may potentially contain better solutions, as opposed to generics which do not use any guidance to direct their operations. Some of these tailor-made low-level heuristics focus on improving the demand satisfaction measures. These heuristics try to satisfy unmet student demands either by constructing new in-class learning activities or by increasing the utilization of already created activities. The remaining tailor-made low-level heuristics focus on reducing the violations of other soft constraints. Each one of these focuses on a specific soft constraint and tries to reduce its violation by destructing a current solution.

The size of this low-level heuristic pool is considerably large, compared to existing hyper-heuristics and also local search methods in the literature, in general. The fact that there are many soft constraints involved in our problem and also the fact that student and teacher solutions are searched in an integrative way during the local search have created the need for developing many low-level heuristics. Naturally, the large size of the heuristic pool makes the task of heuristic selection more important for the effectiveness and efficiency of the search process, requiring an intelligent selection mechanism. Below we briefly explain these low-level heuristics.

- $LLH_1^{generic}$: It randomly picks a student and a time block of a day, and it randomly changes the plan of the picked student to a randomly picked in-class activity that the student demands or to self-study in the picked time.
- $LLH_2^{generic}$: It randomly picks an available teacher in a randomly picked time block of a day, and it randomly changes the assignment state of the picked teacher to either in-class or self-study states in the picked time.
- $LLH_3^{generic}$: It randomly picks a student and a time block of a day, and changes the plan of the picked student to the in-class activity or self-study that results in largest cost reduction in the objective function.
- $LLH_4^{generic}$: It randomly picks an available teacher in a randomly picked time block of a day, and changes the plan of the picked teacher to an assignment state that results in largest cost reduction in the objective function.
- $LLH_5^{generic}$: It randomly picks a student and two different times, and swaps the plans of the picked student in these times.
- $LLH_6^{generic}$: It randomly picks a teacher who is assigned to in-class state in a randomly picked time block of a day and randomly picks another teacher who is a teacher of the course that the firstly picked teacher teaches and is either idle or assigned to self-study state. Then, this heuristic assigns the first teacher to idle state and the second teacher to in-class state in the picked time.
- $LLH_7^{generic}$: It randomly picks a student and a time block of a day, and it randomly changes the plan of the picked student to a randomly picked in-class activity of his/her demand such that the assignment of the student would not require the creation of a new activity session.
- $LLH_8^{generic}$: It randomly picks a student. Then, for assigning the student to an in-class activity that the student demands, it finds a random time where the student is assigned to self-study in which the student can be feasibly assigned to the picked in-class activity.
- $LLH_9^{generic}$: It randomly picks a student and a time block of a day in which the student is assigned to an in-class activity. The aim is to move this assignment to another time. The heuristic randomly selects a different time for the student to be feasibly assigned to the activity that (s)he is assigned in the first picked time. Then, the student is assigned to self-study in the first picked time block.
- $LLH_{10}^{generic}$: This heuristic swaps two students who are assigned to two different sessions, at different times, of the same in-class activity.
- $LLH_{11}^{generic}$: This heuristic swaps two randomly picked students who are assigned to two different in-class activities in a randomly picked time block of a day.
- $LLH_{12}^{generic}$: This heuristic applies a hill-climbing greedy local search on the current solution for seeking improvement opportunities. For each time block of each day it visits the students in a fixed order and assigns the best feasible activity options for them, which will reduce the costs best.
- $LLH_{13}^{tailored}$: This tailor-made heuristic acts on student solutions to reduce the violations of exceeding students' daily course limit. For every student and every day, it reduces one randomly picked in-class activity session of a course from the student in which the student has excess activities assigned to the course on a day.
- $LLH_{14}^{tailored}$: This tailor-made heuristic acts on teacher solutions to reduce the violations of exceeding teachers' weekly in-class assignment limit. For every teacher with an overloaded weekly in-class assignment, it reduces one of the randomly picked in-class assignment from the teacher by assigning him/her to idle state.
- $LLH_{15}^{tailored}$: This tailor-made heuristic acts on teacher solutions to reduce the teacher shortage in the self-study environment. It randomly picks a time block of a day, and assigns a randomly picked idle and available teacher to self-study state.
- $LLH_{16}^{tailored}$: This tailor-made heuristic acts on student solutions to reduce the number of unmet in-class activity demands of students by increasing the utilization of already planned in-class activity sessions. It randomly picks a time block of a day, and for each planned in-class activity in the picked time, it assigns a randomly picked student who is assigned to self-study but with an unmet demand on the activity to the in-class activity, if this assignment does not require the planning of an additional session of the activity.
- $LLH_{17}^{tailored}$: This heuristic is very similar to the previous one. Differently, this heuristic also considers students who are already assigned to some in-class activities at the picked time, for assigning to the activities that have excess capacities.
- $LLH_{18}^{tailored}$: This heuristic also works for increasing the utilization of already planned activity sessions. It randomly picks a time and an activity, then it assigns a random number of idle students, who can be feasibly assigned, to the selected activity at the selected time.
- $LLH_{19}^{tailored}$: This tailor-made heuristic acts on both student and teacher solutions to reduce the number of unmet in-class activity demands of students. It randomly picks a time block of a day and a student who is assigned to self-study at the picked time, and randomly assigns the student to an in-class activity which the student has a demand but is not assigned. Then, it also assigns a randomly picked available teacher, who is qualified to teach the activity and not earlier assigned to in-class assignment state, to in-class state.
- $LLH_{20}^{tailored}$: This tailor-made heuristic acts on both student and teacher solutions to reduce the number of unmet in-class activity demands of students by adding a new in-class activity session. It randomly picks a time block of a day, and plans a

new session of an in-class activity by assigning an idle available teacher who is qualified to teach the activity to in-class activity state and a number of students who are assigned to self-study on the picked time but have unmet demands on the activity to the in-class activity. A session of the in-class activity with the largest potential, the activity that can provide demand satisfaction for the largest group of students, is picked. A number of students are randomly assigned to the picked activity up to the classroom capacity of the activity.

- $LLH_{21}^{tailored}$: This tailor-made heuristic is very similar to the previous one. Differently from the previous heuristic, in the teacher assignment phase, this heuristic also considers non-idle teachers.
- $LLH_{22}^{tailored}$: This heuristic also searches for opportunities to create new in-class activity sessions. For this purpose, it uses an iteration of the batch heuristic.

5. Computational experiments

This section firstly introduces the characteristics of the benchmark instances used to test our approaches. The first experiment focuses on the performance of our method against Gurobi MIP solver. The second experiment presents our benchmark solutions, along with the performance comparisons of the proposed constructive heuristic approaches. The last experiment focuses on the the performance of the local search. This section highlights the important results and patterns of the experiments. For ease of reading, detailed outputs of the experiments are given in Tables 1–7 of the online supplement.

In all of the experiments, an Intel Xeon 2.5 GHz processor with 128 GB memory is used. 50 runs are completed in each non-deterministic setting. Each local search run is limited by one hour of running time and each run is also limited to perform at most 50 000 non-improving iterations. The DTS parameters are offline tuned to the following values: $C^{DTS} = 60,\ \alpha^h = \beta^h = 3, \forall\, h$.

5.1. Data set

This study is an exploratory work for personalized learning models where students master their learning goals at their own pace which results in demand-driven learning activity plans in schools. In our research, we collaborate with the Zo.Leer.Ik! (https://www.zoleerik.nl/) secondary schools network, which currently experiments with various personalized learning models in the Netherlands. There are currently 22 schools in this network. According to the experts from VO-raad (Dutch branch organization for secondary education), several school networks in the Netherlands work on implementing personalized learning. Those networks, including Zo.Leer.Ik!, consist of at least 90 schools in total. Due to the fact that the schools in this network have recently started with personalized learning implementations, sufficient data on student demands are not yet available. We therefore generate artificial instances for this problem that reflect the size-related characteristics of the schools in this network (e.g., number of students, number of teachers, number of classrooms, etc.). The demand scenarios that we consider in our instances are based on an expert's opinions from the network. The student demands in the instances are generated by the consideration of four *demand spread* and two *demand level* scenarios.

Demand Spread: In traditional educational models, students are grouped into fixed age or level groups. In our instances, this concept is again used to consider different demand clusters over the activity set, although in personalized learning there are no longer fixed groups. However in our personalized spread scenarios we consider variety in demands within these conceptual groups.

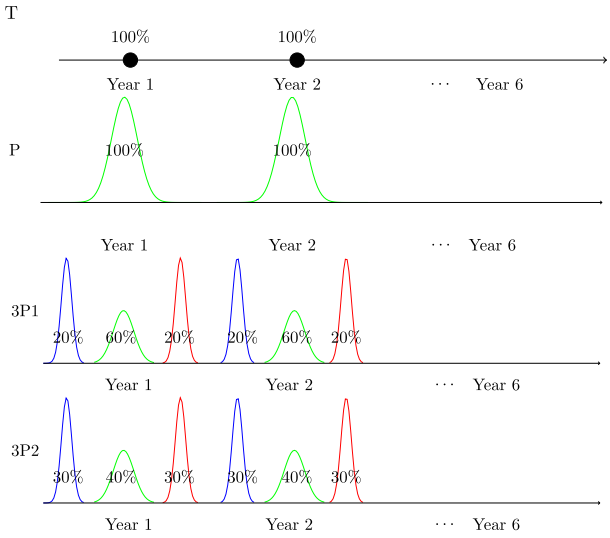

Fig. 3. The illustration of four demand spread scenarios in a course.

The following scenarios are considered for the spread in students' demands.

- T: This is the traditional situation in which the ages determine students' learning paces. This scenario assumes that within an age group there is no variety observed in the students' learning paces. The traditional demand scenario is included in our experiments for the sake of comparing the outcomes of personalized demands with traditional demands.
- P: This is the first personalized demand scenario. It assumes that within an age group, learning demands are spread over activities that range over two learning goals.
- 3P1: This personalized demand scenario assumes that within any age group, three distinct student groups can be observed as a result of their learning speed differences. The three groups represent the slow-, average- and fast-pacing students in each age group. Average-pace groups are assumed to be the largest. The demands of the average speed groups are spread over activities ranging over two learning goals, just like in the first personalized demand scenario. On the other hand, the small groups of slow and fast pacing students demand activities that are spread over only one learning goal.
- 3P2: This scenario is very similar to the previous personalized demand scenario. The only difference of this scenario to the previous one lies in the sizes of learning speed groups within each age group.

Fig. 3 illustrates the differences in these four spread scenarios. This figure shows the learning demands of a representative course in the case of T, P, 3P1 and 3P2 demand spread scenarios. Normal distributions are used for distributing student demands over lessons for the cases that relate to personalized learning demands in our instances. Our instances are in line with the six-year-long secondary education program in the Netherlands.

Demand Level: In traditional models, students use all of their available school time $|B \times D|$ in a week only for course meetings. This is usually between 30 and 40 h in a week. In contrast, in personalized learning, students also learn through self-study activities in schools. This would translate into fewer in-class activity demands. The following demand level scenarios are considered where each student is assumed to be enrolled in six or seven courses.

- 12: Each student demands in total 12 in-class activities per week, two activities per six of his/her courses.

Table 4
Weight parameters.

weight (w^n)	(w_1) Setting 1	(w_2) Setting 2
w^1	1000	1000
w^2	500	500
w^3	300	300
w^4	50	50
w^5	400	2000
w^6	400	2000
w^7	50	50
w^8	300	300
w^9	20	20

- 21: Each student demands in total 21 in-class activities per week, three activities per seven of his/her courses.

School Size: Four school size scenarios are considered : small (S), small_200 (S^{200}), medium (M) and large (L). The medium size reflects the average school in the collaboration network. Other scenarios are obtained by the rough linearization of the medium size (S^{200} instances do not exactly follow this pattern, explanations for this are given in Section 5.2). In fact, S and S^{200} scenarios are not realistic, however they are useful in our experiments to investigate the optimality gaps of our heuristic approach.

- S: 100 students, 12 teachers and 12 classrooms.
- S^{200}: 200 students, 20 teachers and ∞ classrooms.
- M: 800 students, 80 teachers, 40 classrooms.
- L: 2400 students, 240 teachers, 120 classrooms.

Weights of Soft Constraints:

This problem involves nine soft constraints. Our concern in this paper is to use realistic weight settings such that outcomes of our experiments will be also meaningful for providing insights to schools, apart from testing and benchmarking our heuristic approach. To achieve that, we benefit from the results of the two surveys which we conduct with the participation of the school managers of some personalized learning secondary schools in the Netherlands. In the first survey, which is conducted during a workshop session at the "VO-congress, March 28, 2019", 28 participants were given multiple-choice questions that quantify the relative importance levels of given two soft constraints, while in the latter survey seven participants from the Zo.Leer.Ik! network quantified the importance of each soft constraint by directly setting their weights. The results of these surveys almost match with respect to the importance order of soft constraints. However, only one significant difference is realized, the importance level of overloading teachers and students, namely the weights of w^5 and w^6 in the results. The second survey suggests considerably high importance levels for these soft constraints compared to the first survey. As a result, we conduct our experiments by considering two different weight settings, each representing one survey. Table 4 declares these settings; Setting 1 corresponds to the result of the first survey, while the other to the second.

We denote the instance with $x_1 \in \{T, P, 3P1, 3P2\}$ demand spread, $x_2 \in \{12, 21\}$ demand level, $x_3 \in \{S, S^{200}M, L\}$ school size and $x_4 \in \{w_1, w_2\}$ weight setting as "$x^{x_4}_{1_{x_3,x_2}}$" in this document. The details of how these instances are generated are described in the online supplement. The description of our data set in the online supplement also explains the data format of our instances, which can be accessed at https://drive.google.com/drive/folders/1OsJ5CxYNK9lPj8-yqGvn0Nz1ADPqs5lv?usp=sharing.

5.2. Benchmarking against a solver

Our heuristic approach is benchmarked against Gurobi 7.0.2 MIP solver for performance investigation. The small school size instances (S) are used for this investigation, even though they are not realistic cases, because of the limitation of the solver to find optimal or nearly optimal solutions in the considered time limit of four days. In fact, this time limit is too long for practical purposes because in practice a school needs to solve the problem of the upcoming week during the weekend. We compare the Gurobi solutions with the best solutions obtained from the heuristics. The comparisons are given in Table 1 of the online supplement. Firstly, we observe that Gurobi is not able to even find good solutions for the two of the instances: $T^{w^1}_{S,21}$ and $T^{w^2}_{S,21}$. For these instances, the gaps of the Gurobi solutions to the best bounds that Gurobi finds are more than 85% and the heuristic solutions are significantly better than Gurobi's. We argue that the reason for this could be related to the increased numbers of feasible activity group formations of the traditional demand scenario. This also demonstrates the computational challenge of our problem numerically. For the remaining instances, we found that the heuristic solutions have on average 15.72% optimality gap, compared to the best lower bounds found by Gurobi.

In order to experiment with larger instances, some simplifications are made. We use another set of instances which has 200 students (S^{200}) for this purpose. In these instances, the number of classrooms are assumed to be unlimited for each course, all teachers are first-level and also the demand level is only 12. The solver is again given four days of running time. In four out of these eight instances, the heuristic solutions are better in quality then Gurobi solutions. For these simplified instances, the gaps of the heuristic solutions to Gurobi bounds are observed to be significantly smaller compared to the instances with 100 students; the average optimality gap of these instances is 9.19%. Also, for the instances with 3P2 spread scenario, the gaps are even lower than five percent. This could be an indication that the gaps will get smaller as the size of the instances get larger. Although, we are not able to demonstrate this for the larger sizes, such as M school size instances, due to the current performances of the state-of-the-art MIP solvers, we expect that this will be case. Our intuition is that as school size increases, demand satisfaction will get easier because the increasing number of students can be grouped in activities and limited teacher and classroom resources could be used more efficiently, given that the activity set is kept fixed. The detailed explanation for this situation is given later in Section 5.3.2.

5.3. Solutions

This section provides the solutions for the medium and large size instances for benchmarking purposes and also an analysis of two different initialization methods. The detailed performance measures of constructive heuristics are given in Tables 2 and 4, while the solutions of our benchmark instances can be found in Tables 3 and 5 of the online supplement.

5.3.1. Initial solutions

Running Times:

In order to see the running time patterns of the batch and decomposition methods across four demand spread scenarios (T, P, 3P1 and 3P2) in Fig. 4 we present the average running times over all instances of each spread scenario, for each school size. The y-axes of the graphs in this figure give these average CPU values that are measured in seconds. The running times of the batch method are considerably lower compared to the decomposition method in almost all instances, with some exceptions in the case of large size instances. This is expected as decomposition method utilizes the solver for small-scale subproblems. Although these problems are quickly solved, there are 40 of them in our instances. Moreover, the running times seem to be correlated with the demand spread scenarios in both methods. The instances with high-spread scenarios, 3P1 and 3P2, always require more time in both methods. Note

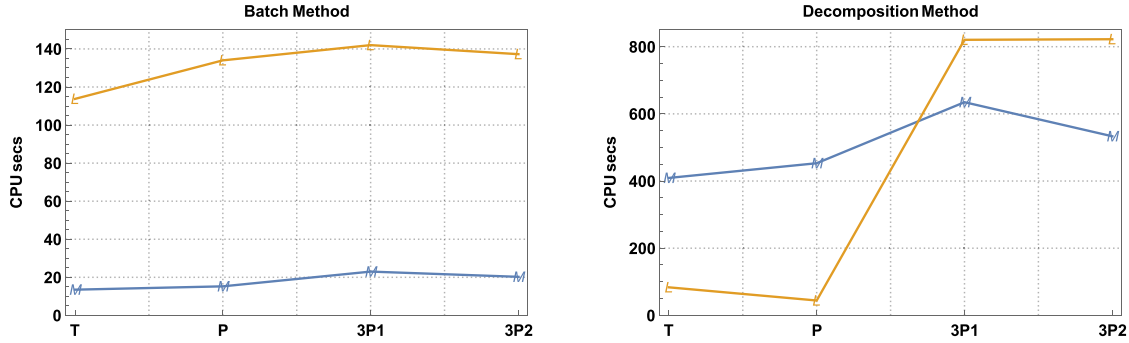

Fig. 4. Average CPUs of batch and decomposition solutions over spread scenarios. M (in blue) and L (in yellow) mark medium and large school size instances, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

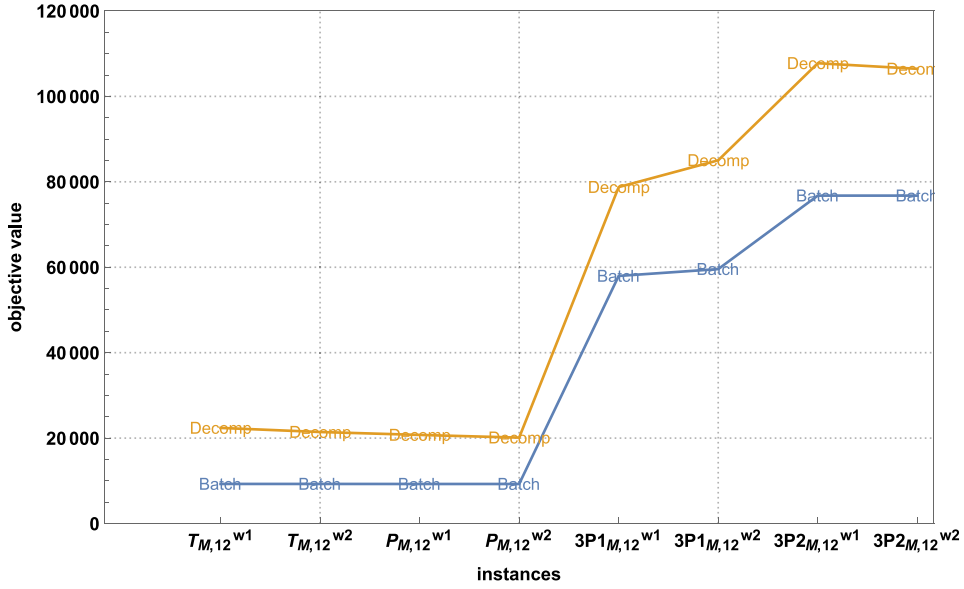

Fig. 5. The objective values initial solutions of medium school size instances with 12 demand level.

Table 5
Overall performance measures of initial solution heuristics in medium and large school size instances (note: each set has 16 instances).

School size	Batch		Decomposition	
M	Ave: 261,724.19	#Best: 9	Ave: 207,407.24	#Best: 7
L	Ave: 253,139.19	#Best 12	Ave: 269,622.38	#Best: 4

that as spread increases, the demanded set of activities will grow which will result in increased variables for activity planning (x_{abd}), increasing the time required to construct the solutions.

Solution Qualities:

Table 5 provides overall performance measures of the batch and decomposition methods, with respect to the qualities of the solutions they produce. "Ave" gives the average objective values of the solutions found by the corresponding method of the instances with the corresponding school size scenario, while "Best" gives the number of times that the corresponding method produces the best-quality solutions in the considered instance set. In both medium and large size instances, the initial solutions produced by the batch method are mostly better in quality than those produced by the decomposition method. However, if the averages of the solutions found by these methods are compared, the decomposition method is better than the batch method in medium school size instances.

The performance of these methods with respect to the qualities of the solutions they produce form patterns over the characteristics of the instances. In order to illustrate these patterns, we present Figs. 5 and 6 which show the objective values found by the two initialization heuristics for the medium school size instances with 12 and 21 demand levels, respectively. It can be observed from these figures that the batch method is always better for the low demand level instances, while in the case of high demand level scenario, the decomposition method is almost always better. We also observe that in the high demand level scenario instances with Setting 2 weights, the decomposition method always outperforms the batch method significantly. The weakness of the decomposition method is that it makes time block based decisions and does not regard the resource availability of the latter time blocks when it is making activity assignment decisions for a block. For instance, this method can make undesirable activity group formations for the sake of satisfying student demands in a time block, although there are better formation opportunities in the future time blocks. In fact, this is the likely reason for the worse performance of the decomposition method in the low demand level instances. In these instances, the opportunities to satisfy student demands in a week are not scarce. Lastly, these experiences with our instances indicate the issue that the instance characteristics plays an important role in the performance of our heuristics for this problem. Therefore, we can not conclude that batch or decomposition method performs best overall. However, based on our computational

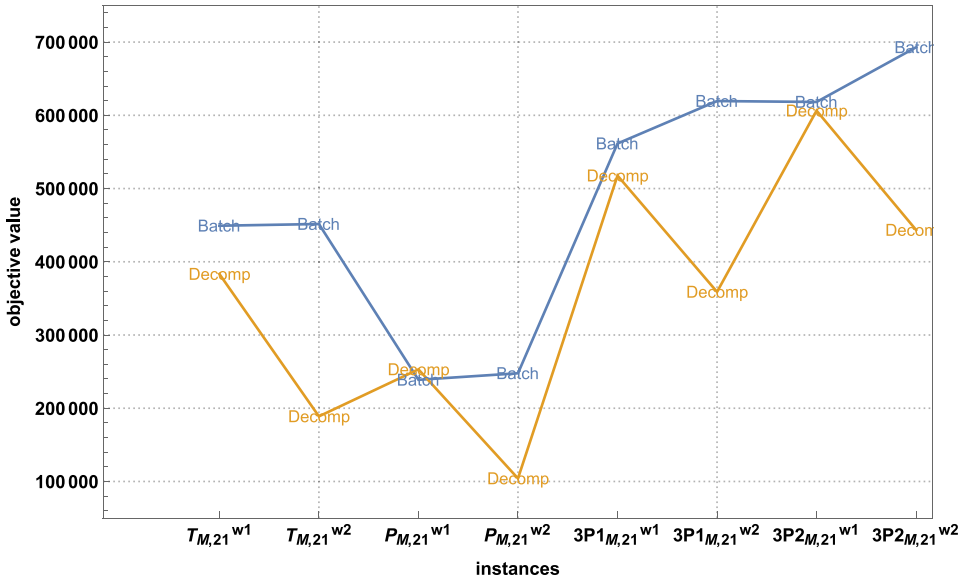

Fig. 6. The objective values initial solutions of medium school size instances with 21 demand level.

Table 6
The average improvement performance of DTSHH in medium and large school size instances.

School size	Batch	Decomposition
M	26.06%	20.01%
L	6.68%	3.40%

experiments, we conjecture that the batch method is more suitable when the resources are not scarce compared to the demands and for the other situations the decomposition method should be used.

5.3.2. Local search applied solutions

Here, the improvement performance of the local search is discussed. Table 6 presents the average improvement percentage by DTSHH for medium and large school size instances, on batch and decomposition initial solutions.

The improvement percentages are always high for the medium school size instances compared to the large size instances. This can be explained with the frequent use of activity creating low-level heuristics, which usually provide high levels of improvement, in medium size instances. On the contrary, in most large school size instances, initial solutions are already able to satisfy most of the demands, leaving less room for the activity creating heuristics. The reason for this is intuitive. In our instances, the number of students and resources increase linearly from the medium to the large school size scenario while the activity set is kept fixed, because in practice the number of courses or lessons within courses usually will not be affected by the size of the school. Given a fixed demand spread scenario, if the number of activities is fixed, increasing the number of students will result in larger activity groups such that the utilization of teachers and classrooms will be higher. In order to also illustrate this numerically, Table 7 is presented.

This table shows the number of unmet demands relating to the batch initial solution and its best found improved solution for the instance with high-demand level, 3P2 demand spread and with large school size. The table shows the results for both weight settings. We first observe the significant quality difference with respect to demand satisfaction levels of the initial solutions of two weight settings. While the improvement percentage is more than 32% in the instance with Setting 2 weights, the setting where the initial solution is significantly poorer with respect to demand satisfaction levels, this percentage is only 1.36% in the same instance with Setting 1 weights. Given the weights of unmet demands (see Table 4), we can calculate and understand that these improvements in the unmet demand levels are the main contributors to a high improvement level of 32%.

We also observe that DTSHH could improve the batch solutions more than the decomposition solutions. In fact, this could also be related to the opportunities to make use of the activity creating low-level heuristics. We mention the weakness of the decomposition method on the ability to consider the available resources of the whole week in the previous section. This method prioritizes the demand satisfaction in each time block, without recognizing the satisfaction chances in the future time blocks of a week. Therefore, the available resources of each time block are depleted greedily. It can be said that in general terms that the decomposition method is greedier than the batch method, which integrates the decisions over the whole week. For this reason, the solutions constructed by the decomposition method can create a significantly larger number of activity sessions compared to those constructed by the batch method and may leave less room for the activity creating heuristics of local search. For instance, in the instance given in Table 7, batch initial solutions create 1952 and 1880 sessions in the Setting 1 and Setting 2 weights, respectively, while decomposition solutions create 3145 and 3167 sessions for the same instances.

5.4. Benchmarking DTS heuristic selection mechanism and low-level heuristics

Our selection hyper-heuristic framework uses the DTS algorithm as an intelligent online-learning heuristic selector. It is of interest to investigate the contribution of this learning mechanism. For this purpose, DTS selection is benchmarked against the non-intelligent mechanism that selects the low-level heuristics uniformly at each iteration, without making use of their historical performances. This framework is named simple random hyper-heuristic (SRHH). Since local search contributes more in the cases of medium school size instances, this benchmarking is performed on these instances. The

Table 7
The improvement of unmet demands with local search for batch solution in a large school size instance.

Unmet Demands	Setting 1		Setting 2	
	Initial	Improved Best	Initial	Improved Best
# unmet high priority demands	81	71	261	111
# unmet low priority demands	348	337	857	543

Table 8
The improvement performances of the low-level heuristics in the local search on the batch initial solution of $T^{w2}_{M.21}$ instance.

low-level heuristic	#calls (ave)	# improved (ave)
$LLH_1^{generic}$	244.38	0.08
$LLH_2^{generic}$	503.72	7.44
$LLH_3^{generic}$	269.64	0.54
$LLH_4^{generic}$	2429.38	91.72
$LLH_5^{generic}$	284.82	1.06
$LLH_6^{generic}$	336.36	4.60
$LLH_7^{generic}$	249.2	0.20
$LLH_8^{generic}$	255.02	0.22
$LLH_9^{generic}$	355.24	2.46
$LLH_{10}^{generic}$	243.1	0.00
$LLH_{11}^{generic}$	243.1	0.00
$LLH_{12}^{generic}$	3808.04	358.28
$LLH_{13}^{tailored}$	242.94	0.00
$LLH_{14}^{tailored}$	262.56	0.42
$LLH_{15}^{tailored}$	479.4	11.36
$LLH_{16}^{tailored}$	348.3	4.98
$LLH_{17}^{tailored}$	407.36	10.06
$LLH_{18}^{tailored}$	258.2	0.72
$LLH_{19}^{tailored}$	248.26	0.08
$LLH_{20}^{tailored}$	298.64	2.90
$LLH_{21}^{tailored}$	336.36	4.60
$LLH_{22}^{tailored}$	242.94	0.00

instance-specific measures relating to DTSHH and SRHH are given in Tables 6 and 7 of the online supplement. ANOVA finds that DTSHH significantly performs better than SRHH on average (with 95% confidence level), on both batch and decomposition initialized solutions. In fact, this difference is already clearly visible, the improvement average of DTSHH is around 10% higher than of SRHH. Clearly, this suggests the important role of the heuristic selection mechanism, keeping in mind that both DTSHH and SRHH uses the same set of 22 low-level heuristics.

We illustrate in Table 8 some performance measures relating to the low-level heuristics on improving the batch initial solution of $T^{w2}_{M.21}$ instance. This instance is deliberately selected as it is one of instances in which the local search improves significantly; thus, we can have a good view of the performances of low-level heuristics. The numbers in this table are the averages of 50 runs. The second column in the table shows the average number of times that low-level heuristics are called in a run, while the third column presents the average number of times that low-level heuristics improved the given solution. The integrative search performed to produce solutions for teachers, students and learning activities in the hyper-heuristic framework led to an inevitably large low-level heuristic pool. With this large heuristic pool, we aim to provide a general local search approach which can adopt itself to different instances of this planning problem. For the selected instance, we can see from Table 8 that almost all of the low-level heuristics are contributing. Table 8 also shows how the online learning DTS selection mechanism adapts itself to exploit the good low-level heuristics in the pool.

The most two successful low-level heuristics for this instance, based on their likelihoods of solution improvement, $LLH_{12}^{generic}$ and $LLH_4^{generic}$, are hill-climbers. This is not striking as hill-climbers greedily aim for improving solutions. Especially, $LLH_{12}^{generic}$, which applies hill-climbing by visiting all students in a fixed order in all time blocks and days, is expected to be successful in improving any given solution.

The low-level heuristics which perturb teacher solutions to improve teacher-related soft constraint violations, $LLH_2^{generic}$, $LLH_4^{generic}$, $LLH_6^{generic}$, $LLH_{14}^{tailored}$ and $LLH_{15}^{tailored}$, are all seem contributing in improving the solutions when applied. Note that our initial solutions are constructed in two phases. The first phase builds a feasible student solution by considering teacher-related hard constraints but is blind to the teacher-related soft constraint violations. These low-level heuristics are expected to reduce the violations of these soft constraints which are ignored during the initial solution construction.

Among the tailor-made low-level heuristics, which try to improve students' demand satisfaction, the ones that try to increase the utilization of already planned activities, without planning new ones, $LLH_{16}^{tailored}$ and $LLH_{17}^{tailored}$, seem more likely to improve a given solution. The low-level heuristics which try to plan new activities need suitable idle teachers and classrooms. Depending on the tightness of available teacher and classroom resources, these low-level heuristics might not have many chances. In the instance given in Table 8, $LLH_{20}^{tailored}$ and $LLH_{21}^{tailored}$ seem to be the most useful ones among these type of low-level heuristics. However, note that all tailored low-level heuristics aimed for demand satisfaction ($LLH_{16}^{tailored} - LLH_{22}^{tailored}$) have the potential to perturb a given solution to a great extent and significantly reduce costs, considering that in general demand satisfaction is the major concern. Even if their improvement chances are low, their impacts are potentially higher.

In addition to having to generate solutions based on student, teacher, and learning activity perturbations, which already requires a reasonably large low-level heuristic pool, it is also important to note that depending on user preferences, the proposed problem can have many weight configurations for the numerous competing soft constraints. In our computational experiments, we benefit from the surveys we have conducted to determine the weight of each soft constraint. Naturally, in practice, different schools can have different weight configurations and instance-specific characteristics, and therefore the initial solutions and the most useful low-level heuristics can change accordingly. For example, in the instances we use in Table 8, the demand level of each student is such that each student only demands three activities in total in a course, while the students' daily course limit is two. Thus, it is very unlikely that a feasible solution of this instance will incur costs for this soft constraint (see SC4). However, if this is not the case, then it might be that the local search would benefit from the low-level heuristic $LLH_{13}^{tailored}$.

We observe that swap type low-level heuristics acting on student solutions, $LLH_5^{generic}$, $LLH_9^{generic}$, $LLH_{10}^{generic}$ and $LLH_{11}^{generic}$, are not very likely to improve a given solution, especially the ones which operate on two selected students, $LLH_{10}^{generic}$ and $LLH_{11}^{generic}$. Note that these swap operators, when applied, will not change the demand satisfaction measures of a given solution, however may

Table 9
The spread challenge for satisfying learning demands.

Unmet Demands (M)	*T*	*P*	3*P*1	3*P*2
% unmet high priority demands	0.82	0.28	2.95	3.59
% unmet low priority demands	0.57	0.27	2.08	2.92
Unmet Demands (L)	*T*	*P*	3*P*1	3*P*2
% unmet high priority demands	0.00	0.00	0.05	0.09
% unmet low priority demands	0.00	0.10	0.18	0.28

change the timing-related soft constraint violations, such as daily course limit violations and planning activities at non-preferred time blocks. Therefore, their potential to improve the objective value of a given solution is bounded by the weight of these timing-related soft constraints.

With these 22 low-level heuristics we aim to provide a solution approach which can generalize well over different instance characteristics. We already have evidence from the two surveys we conducted that soft-constraint violations can be weighed differently in different schools. Each low-level heuristic we propose works for reducing the violations of some specific soft-constraints, either by increasing or decreasing the density of planning decisions. Together, they cover all soft-constraints. With the proposed DTS-based intelligent selection mechanism, our approach can efficiently discover and exploit useful heuristics from the pool, for any given instance.

6. Practical insights and decision support for schools

This section aims to provide practical insights and decision support for schools. Firstly, we discuss the challenge of satisfying high-spread learning demands in schools with limited resources. Later, we provide a sensitivity-analysis based guidance for schools so that they can cope with the process of selecting a suitable weight setting and are able to use our proposed approaches.

6.1. Demand spread challenge

Our experiments on the instances with four different demand spread scenarios indicate the challenge of satisfying demands as they get more spread over the activities. Table 9 illustrates this on the instances with 21 demand level scenario and Setting 2 weights for the medium and large school size scenarios. This table uses the best found heuristic solutions. Note that the pattern seen in this table also is in line with that of Figs. 5 and 6. The table shows that the percentage demand satisfaction levels are significantly worse in the high spread scenarios of 3P1 and 3P2 in both school size scenarios. As the spread of demands increases, the number of students who demand the same activities will decrease. This will lead to smaller activity groups and inefficient use of limited resources. Naturally, it will be more challenging to satisfy high-spread demands. However, we also observe that the demand satisfaction percentages are strictly better for the large school size instance, across all spread scenarios. In Section 5.3.2 we already explain why it is easier to satisfy demands as the school size increases. Our experiments demonstrate the alleviating effects of increased school sizes on the challenge of satisfying spread learning demands. That means that in practice, although it would be always challenging for schools to satisfy high-spread student demands, we expect that large schools can more easily cope with higher levels of spread than small schools.

Table 10
The results of the weight sensitivity analysis on the selected $3P2^{w1}_{M.21}$ instance.

Weights	#u_high	#u_low	# t_ol	#s_ol	#acts	#np_acts	#l_self
w^1, w^2							
{100,50}	231	322	2	0	1235	373	148
{500,250}	199	308	7	43	1274	415	163
{1000,500}	156	290	6	230	1313	440	189
{2000,1000}	166	288	4	363	1275	433	134
{5000,2500}	145	294	4	585	1289	435	148
w^5, w^6							
{50,50}	175	303	6	670	1259	414	134
{250,250}	156	259	5	323	1272	419	144
{500,500}	179	290	5	91	1283	427	146
{1000,1000}	186	290	3	0	1262	413	172
{5000,5000}	186	290	2	0	1261	412	160
w^4							
50	155	291	4	253	1316	444	194
250	178	303	4	249	1257	408	140
500	157	328	2	250	1256	384	127
1000	183	326	2	258	1237	352	139
5000	180	377	2	249	1152	215	111
w^7							
50	155	288	5	231	1306	442	183
250	156	294	4	247	1306	442	182
500	155	295	4	336	1302	438	166
1000	160	304	4	323	1308	442	161
5000	138	345	4	338	1278	430	137

acts: total number of in-class activities planned in a week.
#np_acts: total number of in-class activities planned in non-preferred time blocks in a week.
u_high: total number of unmet high priority demands.
u_low: total number unmet low priority demands.
#l_self: total number of lacking teachers in self-study in a week.
t_ol: total number of overloaded teachers in a week.
#s_ol: total number of student course overloads in a week.

6.2. Weight selection

The weekly flexible demand-driven learning activity planning problem involves numerous quality criteria among which some criteria would be in conflict with some other criteria. In order to test our proposed approaches on realistic situations with respect to the relative importance levels of these criteria, two weight settings are used which are interpreted from the results of two surveys we conducted. However, in practice the issue of which soft constraint is more valuable to satisfy than another will be school specific. In fact, in implementing our methods, the challenging task for schools is the determination of suitable weights for the considered quality criteria. In order to provide guidance for how schools can determine their weights, in this section we illustrate a sensitivity-analysis based method for this selection. In this analysis, we start with a base setting for weights. Then, one-by-one, the weights of some soft constraints are varied while keeping the weights of the others unchanged. For each varied setting, the trade-offs in qualities can be observed.

The results of this analysis are given in Table 10. The results in the table relate to the best solutions found by our heuristics. The insights from our own analysis is summarized as follows.

- Varying the weights of the unmet learning demands: $w^1, w^2 \in \{\{100, 50\}, \{500, 250\}, \{1000, 500\}, \{2000, 1000\}, \{5000, 2000\}\}$. We observe how much we gain with respect to the demand satisfaction and at the same time how much the solution gets worse with respect to other criteria as these weights increase. Specially, we observe that the effect on the number of students who have course overloaded days is significant.
- Varying the overload weights of both teachers and students: $w^5 = w^6 \in \{50, 250, 500, 1000, 5000\}$. The results illustrate the trade-off between satisfying demands and not overloading teachers and students.
- Varying the weight of planning activities at non-preferred times: $w^4 \in \{50, 250, 500, 1000, 5000\}$. The analysis shows how much we will lose with respect to demand satisfaction, if we value the timing of the activities a lot, especially for the case of low priority demands.
- Varying the weight of teacher shortages in the self-study environment: $w^7 \in \{50, 250, 500, 1000, 5000\}$. We observe that if the weight of this soft constraint is too high, then the demand satisfaction levels will get affected significantly, again especially the low priority demands.

As can be seen above, this analysis can shed light into how the quality measures change as weights vary. In practice, if a school would like to use this analysis to select their weights, the school might also need to select the order in which the weights are varied. For instance, the school can start with varying the weights of the soft constraints that they find most important. Then, they can fix these weights to the level that they think gives the most desirable outcome. Weights of all soft constraints can be fixed one-by-one in this fashion.

7. Conclusion

In this work, we explore the weekly activity planning problem of personalized learning models in which students master their learning goals at their own pace with their own methods. This is a student demand-driven flexible planning problem which involves decisions on students, activities, teachers and classrooms. We exploit the flexibility in planning to reduce some of the decision layers and provide a reduced MILP model. However, this exploitation is not sufficient for standard solvers to find optimal solutions efficiently to real-life sized problem instances. Alternatively, we provide a heuristic approach which firstly builds greedy initial solutions with batch and decomposition methods and then improves these solutions with a dynamic Thomson sampling based hyper-heuristic framework. We use a relatively large low-level heuristic pool. Our hyper-heuristic framework integrates the dynamic Thompson sampling algorithm (DTS), an algorithm proposed for the dynamic multi-armed bandit problem, as an intelligent heuristic selection mechanism. We demonstrate the applicability of our methods in the context of Dutch secondary schools, for which we make use of expert opinions and survey results in generating benchmark instances. Our experiments illustrate the computational challenge of this problem and demonstrate the effectiveness of the proposed heuristic methods. Specifically, we show under which circumstances which of our constructive heuristics perform better. Moreover, we numerically demonstrate the challenge of satisfying high-spread demands in schools, and also the alleviating effects of increasing school sizes in this manner.

Acknowledgment

The authors thank Wim Kokx, president of the board of Openbare Scholengroep Vlaardingen Schiedam, for providing expert insights and feedback.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2020.03.038

References

Ahmed, L. N., Ozcan, E., & Kheiri, A. (2015). Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications, 42*, 5463–5471.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., & Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The greek case. *Computers and Operations Research, 35*(4), 1265–1280.

Bray, B., & McClaskey, K. (2013). A step-by-step guide to personalize learning. *Learning & Leading with Technology, 40*(7), 12–19.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. (2009). A classification of hyper-heuristic approaches. *Technical Report*. University of Nottingham. Computer Science Technical Report No. NOTTCS-TR-SUB-0906241359-0664.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research, 176*, 5463–5471.

Cheng, E., Kruk, S., & Lipman, M. (2002). Flow formulations for the student scheduling problem. In E. Burke, & P. De Causmaecker (Eds.), *Practice and theory of automated timetabling IV. PATAT 2002. Lecture notes in computer science: 2740* (pp. 299–309). Springer-Verlag.

Cordeau, J.-F., Toth, P., & Vigo, D. (1998). A survey of optimization models for train routing and scheduling. *Transportation Science, 32*(4), 380–404.

Da Costa, L., Fialho, A., Schoenauer, M., & Sebag, M. (2008). Adaptive operator selection with dynamic multi-armed bandits. In *Genetic and evolutionary computation conference (GECCO), ACM: 38* (pp. 913–920).

Dorneles, Á. P., de Araújo, O. C. B., & Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers and Operations Research, 52*, 29–38.

Eiken, O. (2011). The kunskapsskolan ("the knowledge school": A personalised approach to education. In *CELE Exchange, Centre for Effective Learning Environments, OECD Publishing, Paris: 1* (p. 6).

Fialho, A., Da Costa, L., Schoenauer, M., & Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence, 60*, 25–64.

Fonseca, G. H. G., & Santos, H. G. (2014). Variable neighborhood search based algorithms for high school timetabling. *Computers and Operations Research, 52*, 203–208.

Fonseca, G. H. G., Santos, H. G., Carrano, E. G., & Stidsen, T. J. R. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research, 262*, 28–39.

Gupta, N., Granmo, O., & Agrawala, A. (2011). Thompson sampling for dynamic multi-armed bandits. In *10th international conference on machine learning and applications*.

Kannan, A., van den Berg, G., & Kuo, A. (2012). iSchedule to personalized learning. *Interfaces, 42*(5), 437–448.

de Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research, 182*(2), 481–501.

Kristiansen, S., Sørensen, M., & Stidsen, T. R. (2011). Elective course planning. *European Journal of Operational Research, 215*(3), 713–720.
Phillips, A. E., Waterer, H., Ehrgott, M., & Ryan, D. M. (2015). Integer programming methods for large-scale practical classroom assignment problems. *Computers and Operations Research, 53*, 42–53.
Pillac, V., Gendreau, M., Gueret, C., & Medaglia, A. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1–11.
Pillay, N. (2014). A survey of school timetabling research. *Annals of Operations Research, 218*, 261–293.
Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals of Operations Research, 239*, 3–38.
Pillay, N., & Banzhaf, W. (2009). A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. *European Journal of Operational Research, 197*(2), 482–491.
Prain, V., Cox, P., Deed, C., Dorman, J., Edwards, D., Farrelly, C., … Yager, Z. (2013). Personalised learning: lessons to be learnt. *British Educational Research Journal, 39*(4), 654–676.
Santiago-Mozos, R., Salcedo-Sanz, S., DePrado-Cumplido, M., & Bousoño–Calzón, C. (2005). A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a spanish university. *Computers and Operations Research, 32*(7), 1761–1776.
Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review, 13*, 87–127.
West-Burnham, J., & Coates, M. (2005). *Personalizing learning: Transforming education for every child* (first ed.). Network Continuum Education.